(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 December 2001 (06.12.2001)

PCT

(10) International Publication Number
WO 01/93021 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/IL01/00210

(22) International Filing Date: 5 March 2001 (05.03.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/585,685 1 June 2000 (01.06.2000) US

(71) Applicant *(for all designated States except US)*: **ADUVA INC.** [US/US]; 19731 La Mar Drive, Cupertino, CA 95014 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **SEGAL, Uri** [IL/IL]; 16 Lipsky Street, 62195 Tel Aviv (IL). **TE'ENI, Moddy** [IL/IL]; 3 Wiezel Street, 64241 Tel Aviv (IL). **SEGAL, Harel** [IL/IL]; 5 Hanna Snee Street, 51586 Bnei Berak (IL).

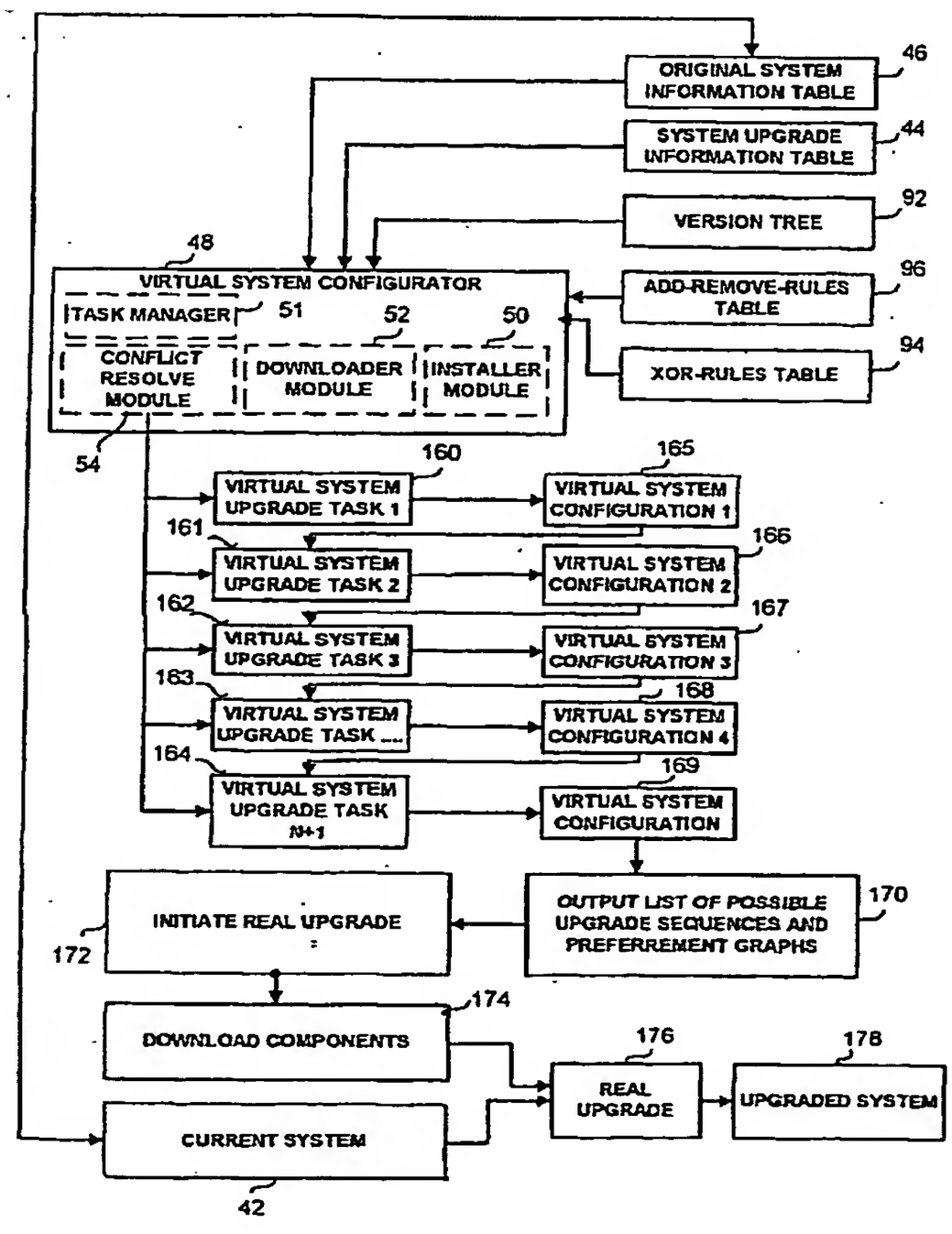
(74) Agents: **AGMON, Jonathan** et al.; Soroker - Agmon, Advocates and Patent Attorneys, 12th Floor, Levinstein Tower, 23 Petach Tikva Road, 66184 Tel Aviv (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: A VIRTUAL SYSTEM CONFIGURATOR FOR CLIENT SYSTEMS



(57) Abstract: A system and method designed for computer system upgrade support that performs a virtual upgrade process of software components. Dependency checking and automatic conflict resolving features are supported. Subsequent to the result of the virtual process, an operative, real system upgrade is achieved. The virtual application utilizes a comprehensive rules language and a dynamically growing software component information base to accomplish the installation of new or improved components into an original system.

WO 01/93021 A2

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A VIRTUAL SYSTEM CONFIGURATOR FOR CLIENT SYSTEMS

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to apparatus and method for supporting computer system management, in general, and for providing assistance to system managers and users of client systems in a computing environment in the process of installing and uninstalling software components, in particular.

DISCUSSION OF RELATED ART

A computer software system is modular in the sense that it consists of a plurality of distinct software programs that run in close cooperation. Modern state-of-art computer systems consist of a remarkably complex set of functionally interconnected software programs therefore installing or removing software in a computing environment is a demanding task.

Following the accelerated progression in computer-related technologies, software development is fast-paced, so new software and new versions of old software are announced continually.

New users are being constantly exposed to computers and computer systems. The majority of these users are not computer professionals. Therefore, computer system management to a great extent is being done by persons whose expertise in the internals of computer operating systems is hardly sufficient to deal with the complexities of a necessary system upgrade or with an urgent installation of a new set of hardware device drivers.

The specific operation of altering a software configuration of a computer system generally referred to as "upgrading". Upgrading a computer system involves installation, removal, replacement of software components or any combination thereof.

Typically, upgrading a computer system involves either adding new software components to the ones already in place or replacing existing software components by recently developed or improved versions thereof. The process involves retrieval of the appropriate components from one or more sources (commercial software companies, independent developers, distribution sites on the Internet or any combination thereof) and installing the components by means of appropriate utility programs. The installation utilities could be part of the original computer system or could be supplied by the vendor and/or the developer.

The majority of the available installation utilities operate in a basic fashion. The components of a new type to be installed are inserted and added to the computer system whereas new versions of existing component types overwrite the previous versions rendering them thereby inoperative. The computer system configuration files are updated accordingly and the original configuration files are removed. Installation utilities of a more advanced type include some operational safety measures such as storing the deleted components in specific storage areas, saving original configuration files and providing a list of actions performed.

There are often interdependencies across the installed components; a particular component or a particular version of one component needed to run an another component or a particular version of another component. Some specific utilities check for inter-component dependencies and provide warnings about possible incompatibilities between the original system configuration and the set of components about to be installed. In the face of these unfavorable circumstances, the utility programs typically display one or more warning messages, discontinue the upgrade process and permit the system manager to solve the problem in a conventional manner. Solving the problem in such a manner may require a extended time-period as it may involve the reading of a large amount of documentation before proceeding, a plurality of requests for software support from the appropriate vendors, waiting for answers from the corresponding vendors and the like.

Although a number of yet more advanced products offer automatic installation of software components with dependency checking and conflict resolving such as Update Agent of Red Hat Linux and APT (A Package Tool) of Debian, the solution proposed thereby is not comprehensive enough. Both products have a number of disadvantages such as handling Software components only thereby ignoring the vital inter-dependency issues between Software, Hardware and operating system Kernel components. Furthermore, both products have a relatively static Component Knowledge Base, a limited set of dependency rules and a restricted set of user options.

It would be evident to those skilled in the art that considerable technical knowledge and expertise is needed for the computer system upgrade although some of the requisite steps are quite straightforward and satisfactory means is provided for the assistance thereof. In contrast, for the most difficult part of the upgrade process i.e., for configuring the system, no comprehensive automatic tools exist yet.

It will be also obvious to those skilled in the art that there is a long felt need for a comprehensive, automated system management support conducive to fast, solid, efficient and convenient system upgrade. There is an urgent need specifically for an advanced service designed to aid and assist system managers and users of small-size client systems in the exacting tasks related to the maintenance of complex and dynamically evolving contemporary computer systems.

SUMMARY OF THE PRESENT INVENTION

One aspect of the present invention regards a method of supporting system management by providing assistance to users of computer platforms running client systems with a virtual system configuration application for the virtual upgrade of software elements in a computing environment wherein virtual system configuration software is implemented to execute a system upgrade with inter-component dependency checks and inter-component conflict resolving, utilizing a software component knowledge base to be used as information source to said dependency checks and conflict resolving. An original system configuration application is also provided to build an original system configuration information base. A storage device associated with the computing platforms holds the application and tables required for the accomplishment of the virtual system upgrade, an I/O device is utilized as the user interface for the activation of the application and a CPU is operative in the execution of said application.

A second aspect of the present invention regards a system that is supporting a virtual system configuration application for the execution of a virtual system upgrade whereby software elements of a computer system are upgraded using modules adapted to perform inter-component dependency checks and automatic conflict resolving. The system comprises storage means to hold the necessary information and modules, I/O means to enable communication between a user and the system, CPU means to provide for processing of the information, virtual system upgrade means such as software modules operative in checking the potential conflicts among the upgraded software components, original system configuration creation means to enable said user to check dependencies and conflicts within the existing system and information base creation and downloading means to provide the requisite control data as input to the operations of the virtual upgrade process.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of the computerized environment in which the virtual system configurator is implemented in accordance with a preferred embodiment of the present invention;

Fig. 2 is a simplified block diagram of the configuration relevant to the implementation of the virtual system configurator on the client device in accordance with a preferred embodiment of the present invention;

Fig. 3 is a simplified block diagram showing the various options the user of the client device can exercise pertaining to the implementation of the virtual system configurator;

Fig. 4 is a simplified block diagram showing the process of a full system upgrade on the client device as performed in accordance with a preferred embodiment of the present invention;

Fig. 5 is an illustration of the modules, tables and operations pertaining to the performance of a system upgrade on the client device.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiments of the present invention are in the context of the Linux environment. Linux is a full-featured UNIX-like computer system that was designed to provide personal computer users a free or very low-cost operating system comparable to the traditional and usually more expensive UNIX systems. Linux has a reputation of a very efficient and fast-performing system. Linus Thorvald of the University of Helsinki in Finland has created the Linux kernel, the central part of the operating system. To complete the operating system, Linus Thorvald and other team members made use of system components developed by members of the Free Software Foundation for the GNU project

As Linux conforms to the POSIX standard user and programming interfaces, developers can write programs that can be ported to other operating systems. Linux is distributed commercially by a number of companies.

As the source code was made freely available and further development by others was widely encouraged, a massive, worldwide development is pursued today by a great number of companies, programmer teams and individuals independently or in cooperation with others. As a result of this extensive activity, a multiplicity of new drivers, modules, packages, and applications are written and distributed for Linux constantly. The new systems, versions, and components written for Linux are all potentially installable and operative in every Linux installation. Linux is a powerful and popular operating system and it is widely used on personal computers. It will be readily perceived that a proposed solution designed to support the management of computer client systems will be particularly suitable for the Linux environment.

It will be also understood that the proposed solution may work in other computing environments, under diverse operating systems (O/S) and on varied hardware platforms.

The present invention overcomes the disadvantages of the prior art by proposing a novel method and a system, which provides an automated or a semi-automated mechanism for the purpose of altering a specific software

configuration in a computer client system by modifying the combination of software components said system consists of. The present invention provides an advanced method and system that simplifies and the automates of the upgrade process when installing components, thereby practically eliminating the possibility for incompatibilities to occur in the course of the upgrade process in an O/S environment. To achieve its objectives the present invention proposes the use of a Virtual System Configurator Client System.

A central server-based knowledge base is created and maintained. The knowledge base ideally includes the entire set of available O/S-compatible software entities such as kernels, utilities, compilers, packages, patches, device drivers, configuration files and the like. All the above mentioned entities will be referred to as "component objects" in the text of this document. Typically, component objects are associated with information units describing the component objects' characteristics. The information units are received along with the component objects or could be retrieved separately. Component information will be referred to as "component data" in the text of this document.

The appropriate components and component suppliers thereof are identified. The suppliers could be software companies, individual developers, Internet software distribution sites and the like. The components and the component data associated therewith are collected, retrieved and stored into the memory device of the central server device using various electronic transmission means such as downloading from an Internet site. The component supplier, component object and component data identification process, the component object and the component data collection and reception is an ongoing operation on the server device. Therefore the sets of the assembled component objects and related component data on the server device are being constantly extended and continuously updated. Using related component data the component objects and relationships thereof are interrogated or researched. The results are encoded and stored on the storage device of the server system in a structured pattern onto a specially designed knowledge base, which is subsequently validated by repeated

testing of the component relationships. The research and validation processes are also continuous. Consequently, at any given point in time the knowledge base consists of state-of-art component objects and an up-to-date abstract model of all the existing interdependencies thereof. Typically, the component object files include the most recent component objects available and the latest versions of the existing component objects.

Users of client systems operating in a computing environment, under the control of an O/S such as Linux and requiring a system upgrade connect intermittently to the central server-based knowledge base. The client system provides a virtual system configurator application that enables the users to download the knowledge base from the server, resolve the conflicts resulting of the virtual system upgrade, download and install the pertinent component objects and perform a real system upgrade. The process can be completed either automatically, semi-automatically, or manually based upon user input.

Referring now to the drawings, in which like numerals represent elements throughout the several figures, aspects of the present invention and the exemplary operating environment will be described.

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the present invention may be implemented. While the invention will be described in the general context of application programs that run on a client-server system, those skilled in the art will recognize that the invention also may be implemented in combination with other environments.

Referring now to Fig. 1 there is shown constructed and operative in accordance with the preferred embodiment of the present invention the computing environment in which Virtual System Configurator Client system is operating. The components 12,14,16,18 are retrieved entered into the server system 20, collected on a storage device on the server system 20 and processed by the component handlers 22. The component objects 24 are stored into the component

object files 30 and the associated component data 26 is inserted into the Component Knowledge Base 28. The component handlers 22 also perform research and validation procedures on the component data 26 stored in the knowledge base 28. Using an expressly designed rules language, the functional dependencies of the component objects are encoded into an abstract model representing inter-component dependency rules. The encoded rules are inserted into the knowledge base 28 creating new entries or updating the suitable entries of the knowledge base 28.

The client system 40 is intermittently connected to the server system 20. The client system 40 consists of a storage device 41, a Central Processing Unit (CPU) 43, an Input/Output (I/O) device 45, an operating system 56, an original system information table 46, a system upgrade information table 44, a Virtual Upgrade module 48, a Conflict Resolver module 54, a Downloader module 52 and an Installer module 50. In the preferred embodiment of the invention, the operating system 42 is a Linux kernel-based system. The original system information table 46 holds information regarding the components installed on the current system thereby mirroring the current system configuration. The system upgrade information table 44 comprises the list of component objects to be installed, uninstalled or upgraded by user request. The virtual upgrade module 48 performs the requested virtual upgrade utilizing the component knowledge base 28 of server system 20. Virtual upgrade module 48 activates conflict resolver module 54 to resolve the conflicts induced by the virtual installation of the requested component objects. Conflict resolver 54 using the dependency rules retrieved from the component knowledge base 28, performs the task thereof in conjunction with the system upgrade information table 44. One or more virtual configuration files are created and tested. If the resulting virtual configuration files are conflict-free the installation of the components is confirmed, the downloader module 52 downloads all necessary component objects from the component object files 30 of server system 20 and the installer module 50 installs the component objects onto the operating system 42.

The server system 20 provides the information necessary for a virtual system upgrade according to the client system 40 requests through the component knowledge base 28. The server system 20 also provides the component objects needed for a given upgrade request. The server 20 further provides diverse services for the client system 40.

It will be appreciated by those skilled in the art that there are other ways to implement the proposed invention. In other embodiments of the invention, different tables and operations could be utilized. The present invention is not limited to the specific mode of operation presented.

Referring now to Fig. 2 there is provided a more illustration of the client system 40 software configuration. The client system 40 contains an operating system 42, an original system information table 46, a system upgrade information table 44, the user tables 80, the component knowledge base 90, the virtual system configurator 48 and the component object files 99. The original system information table 46 is a data structure on the storage device 41 of Fig. 1 of client system 40 containing kernel configuration information 72, hardware configuration information 74, software configuration information 76 and the user options 78. The original system information 46 holds information about the components installed in the operating system 42 thereby mirroring the structure of operating system 42. Original system information 46 is created by the user of client system 40 or could be created automatically by the virtual system configurator 48 application. Typically, in the client system 40 most operations related to the virtual system configuration process could be performed automatically or semi-automatically where every phase of the process requires the confirmation or the initiation of the user utilizing the client system 40.

System upgrade information 44 is a data structure on a storage device 41 of client system 40 holding the list of components to be installed, uninstalled or updated in a requisite upgrading process. System upgrade information table 44

is created by the user of the client system 40 or could be created automatically according to the user's preferences by the virtual system configurator 48.

The user tables 80 are data structures on the storage device 41 of client system 40. The tables consist of a user preferences table 84. The user of client system 40 builds and updates the tables. The user preferences table 84 consists of entries determining various user options related to the operation of the virtual system configuration process. A detailed description of the above mentioned options will be set forth hereunder in association with the following drawings.

The component Knowledge Base 90 is a collection of interlinked data structures on the storage device 41 designed to provide control information for the virtual system configuration process. The component knowledge base 90 consists of the version tree 92, the xor table 94, the add-remove table 96 and the links table 98. The component knowledge base 90 is a copy or a partial copy of the component knowledge base 28 created and maintained on server system 20 and downloaded automatically or by user request to the storage device 41 of the client system 40. The component knowledge base 90 is supports the process of resolving conflicts arising from inter-component dependencies induced by the installation of the requested components into client system 40 operating system 42. The version tree 92 consists of component data records associated with the component objects that are stored in component object files 99. The component data describe component object characteristics or component object attributes. For each component object one or more component data record exists in the version tree 90. There are several types of components such as hardware type, kernel type, software type and the like. Each type could have several subtypes for example; a component of the hardware type could have as subtypes; ISA (Industry Standard Architecture) cards, ISA PnP (Plug and Play) cards, Serial ports and the like. Each subtype in turn could have several subtypes of its own for example a hardware type component ISA cards subtype could have subtypes of its own like: Ethernet, SVGA (Super Video Graphics Array), Sound, Modem and the like. In the

preferred embodiment of the invention, version tree 92 is organized hierarchically i.e., in a tree-like manner and held in a tree-like data structure.

Xor table 94 and add-remove table 96 are copies or partial copies of identical tables included in component knowledge base 28 on server system 20, created and maintained on server system 20 and downloaded automatically or by user request to a storage device of client system 40. Xor Rules table 94 and Add-Remove Rules table 96 consists of rules provided for checking the dependencies between one or more component objects. The rules records in the Xor rules table 94 and the add-remove rules table 96 are created and maintained utilizing a specific rules language. Xor Rules table 94 consists of two fields: (1) a rule index and (2) a component number. Add-remove rules table 96 consists of three fields: (1) a rule index (2), a "has" field, i.e., the system has the component that appears in the field and (3) a "need" field, i.e., the system needs the component appearing in the field.

Links table 98 is a copy or a partial copy of an identical table included in component knowledge base 28 on server system 20. Links table 98 provides links between different leaves of the version tree 92. The purpose of the links is to connect two groups of components; for example, a kernel version and a fixing patch thereof. The fields of links table 94 are (1) a link index, (2) id of the first node to be connected, (3) id of the second node to be connected, (4) a link type and (5) a version stamp.

Component objects file 99 consists of component objects. Component objects file 99 is a copy or a partial copy of component objects file 30 of server system 20. Component objects file 30 is downloaded automatically or by user request from server system 20 for the actual system upgrade process.

Virtual System Configurator 48 is the application program designed to execute the requested upgrading process in its entirety. Virtual System Configurator 48 contains a number of vital modules such as the Task Manager 51, Conflict Resolver 54, Downloader Manager 52 and Installer 50. The operations

and the functionalities of each module will be described hereunder in association with the following drawings.

It will be appreciated by those skilled in the art that in a different embodiment of the present invention different tables could be used. The present invention is not limited by the specific description presented here.

Referring now to Fig. 3 there is shown a simplified block diagram showing the various options the user of the client device can exercise pertaining to the implementation of the virtual system configurator in accordance with a preferred embodiment of the present invention. The user of client system 40 may enter commands and information into the computer through a MMI (Man Machine Interface) in association with a monitor device and an I/O device utilizing standard interfacing means such as a keyboard device, a mouse device and the like. The various operations the user might initiate and the various tools he might utilize will be described next.

The user maintains the original system information table 46 of Fig. 2 (box 102), initiates checks for conflicts on the original system (box 106), maintains user tables 80 of Fig. 2 (box 108), sends queries to the server system 20 of Fig. 1 (box 110), requests downloads of component sources (box 112) and component objects (box 114) from the server system 20 of Fig. 1, builds system upgrade information table 44 of Fig. 2 (box 116) and initiates virtual system upgrade (box 118).

The original system information table 46 of Fig. 2 is created every time a system upgrade is under consideration. The building of original system table 46 of Fig. 2 (box 120) is performed in three stages: (1) hardware configuration determination performed by utilizing standard functions of the operating system or accessing particular files in specific libraries such as /lib/pci/lothar that hold the relevant information, (2) software configuration determination by accessing particular files such as RPM DB that hold the relevant information, and (3) kernel configuration determination applying functional tests and checking the kernel

header files. Therefore, the original system information table 46 of Fig.2 will consist of data about the entire component set of the current system. After building the original system information table the user might query the entries of the table (box 122) and insert component-specific preferences (box 124). By default in component is marked as locked. By user request each component might be marked as: (1) locked (uninstall disabled), (2) unlocked (uninstall enabled), and (3) information request (a request to server system 20 of Fig. 1 to send specific information such as technical tips of news about the component marked). Therefore when the virtual system configurator 48 of Fig. 2 will be activated, components marked as "locked" will not be removed or replaced. For example, the user could mark the kernel itself as a locked component thereby preventing the virtual system configurator to apply any changes to the kernel. Thus, the user has complete control of the upgrade process.

The user might check conflicts existing in the original system (box 106) by activating the virtual system configurator 48 of Fig.2 in a special mode of operation. The original system information will be checked and the user will be informed of the existing conflicts in the current system caused by non-matched components or non-matched version of components.

The maintenance of the user tables 80 of Fig. 2 (box 108) is discussed next. In the preferred embodiment of the invention, the user tables 80 of Fig. 2 are stored on a storage device of client system 40 of Fig.1. It will be easily perceived that the user tables 80 of Fig. 1 could be stored on a storage device of server system 20 of Fig. 1 as well. The user of client system 40 of Fig. 2 creates and maintains a user profile table 82 of Fig. 2 (box 126) that might contain useful data such as the user type. The user type might indicate whether the user operates a command-line mode display or a graphical mode display. According to the data contained in the user profile table 82 of Fig. 2 the application might take automatic decisions such as what type of components to download to client system 40 of Fig. 1 and the like. The user is allowed to update the mode of

operation (box 128) thereby deciding if the application executes automatically or in distinct stages at the end of which the user is required to confirm the continuation of the process. Some other preferences the user might want to set are instructions to the server system such as "apply kernel patches automatically" (box 128), "send notification about new versions for specific components automatically" (box 130) and "download all new version for specific components automatically" (box 132).

It will be easily perceived by one skilled in the art that in other embodiments of the present invention a plurality of additional preferences could be utilized and more useful data fields could be added to the user tables.

Another option the user of client system 40 of Fig. 1 can practice is requests to server system 20 of Fig. 1. Server system 20 could be ordered to download sources of specific components (box 112) or download specific component objects (box 114). The user could also send to server system 20 various queries (box 110) relating to new components, new versions of components or specific component data.

The user manually initiates Virtual System Upgrade application 48 of Fig. 2 (box 118). According to the user preferences table 84 the initialized process can be automatic or can be closely supervised by the user (box 134) and restarted manually at every stage. The user can also manually initiate a real system upgrade (box 136) independently or following a virtual system configuration.

It should be understood that some of the features described above could be turned off and other features could be added. In an another embodiment of the invention for example, the real system upgrade could be initiated manually only.

Referring now to Fig. 4 that illustrates the process of a system upgrade on the client device. The process could run in a fully automatic mode without user interaction or in a semi-automatic mode where the application is closely

interacting with the user. When the process running in an automatic mode execution can be aborted by the application for various reasons that will be described hereunder. Naturally, in the semi-automatic mode of operation a partial process is allowed i.e., only some of the phases could be performed as the user can stop the process at any stage. When in semi-automatic mode under user control, the user checks the results of the prior stage and initiates the next stage manually.

At step 140, the original system information table 46 of Fig. 2 is built. The original system table 46 consists of the entire set of component data representing the current system including hardware, software and kernel elements before the contemplated installation. At step 142, the system upgrade information table 44 of Fig. 2 is created. System upgrade table 44 contains a set of requests for installing or uninstalling components. Typically, system upgrade table 44 is created by the user of client system 40 utilizing the MMI in association with a monitor device and an I/O device through which the user inputs the appropriate commands. It should be clear that more than one system upgrade table 44 could be created. System upgrade table 44 could also be built automatically by the virtual system configurator 48 application as a preparation for the automatic process such as the installation of kernel patches. At step 144 the user of client system 40 could insert component-specific preferences into original system table 46 such as "locks" a particular components to prevent the removal thereof and the like. In preparation for the virtual system upgrade the user could manually request from the server system 20 to download all or a part of the component knowledge base 28 of Fig. 1 to create the component knowledge base 90 on the client system 40. In a different mode of operation, the component knowledge base 28 could be downloaded automatically by the virtual system configurator 48 application at a later stage.

At step 148, the user manually activates the system upgrade task manager 51 that will manage the process of virtual system configuration. A more detailed explanation of the functionalities and operations of virtual system

upgrade task manager 51 will be described hereunder associated with the following drawings. At step 150 virtual upgrade task manager 51 will output a list of possible actions for the user to perform in order to accomplish a seamless real system upgrade. The list could give direct instructions or could consist of various options and/or recommendations. The list could also recommend not performing a system upgrade for diverse reasons such as unresolvable inter-component dependency conflicts among the various requested components to be installed or among the components already in the system and the components to be installed. Accordingly, the user could select a preferred action at step 152 and request the downloading of the required components from component objects file 30 on the server system 20 thereby creating the component objects file 99 on the client system 40 at step 154. If the necessary component objects exist on the server system 20, the requested download is performed. Subsequently at step 156, the system is upgraded and if booting the system is indispensable, the system boot is performed at step 158.

It must be emphasized that in accordance with the present invention all the steps described above could be performed automatically in an uninterrupted sequence without user intervention. It also should be clear that in the case of unresolvable inter-component dependency conflicts the automatic process will terminate and the responsibility for the decision-making passes to the user of the client system 40.

Fig. 5 is an illustration of the modules, tables and operations pertaining to the performance of a full-featured system upgrade on the client device, in accordance with a preferred embodiment of the present invention. The Virtual System Configurator application 48 activated manually by the user of the client system 40 or automatically according to the mode of operation 128 of Fig. 3. The input for the virtual system configurator 48 is discussed next. The original system information table 46 comprises the list of the components installed in the current system 42. The system upgrade information table 44 comprises a list of actions

related to the set of components to be installed, uninstalled or updated in order to create a new upgraded system 178. The version tree 92 downloaded from server system 20 of Fig. 1 comprising component data and useful pointers, the add-remove rules table 96 and the xor-rules table 94 also downloaded from server system 20 comprising the dependency rules intended to be utilized by the conflict resolve module 54 of the virtual system configurator 48.

The Virtual System Configurator 48 comprises a number of key modules such as the Task Manager 51, the conflict resolve module 54, the downloader module 52 and the installer module 50. The Task Manager 51 is a collection of tasks and a set of operations on tasks. The Task Manager 51 creates and activates a number of tasks to run concurrently. Typically, all tasks have the same functionalities. The Task Manager 51 creates a task such as tasks 160,161,162,163,164. A task is controlled by the user of the client system 40 through the MMI associated with a display device and an I/O device via the Task Manager 51. A task can be stopped, re-started, re-directed to re-start at a different phase of execution and cancelled. The life cycle of task consists of task-phases. Each task-phase initiated by the Task Manager 51 command. The descriptions of the task-phases are as follows:

(1) The Task Manager 51 creates a task

(2) The Task Manager passes the task information about a components or group of components to be installed, uninstalled or updated. The information could be structured as a list of commands such as "upgrade editor, uninstall icq. install graphic mail element."

(3) The Task Manager 51 commands the task to resolve dependency conflicts. In order to execute the command the task has to find or download the set of tables needed to handle the virtual installation of the specific component.

(4) Task manger commands the task to enter "current" state and task begins "preparing" i.e., downloading the required tables.

(5) The Task Manager 51 commands the task to run. Task calls installer module 50.

(6) The task is completed. The task sends a message to the Task Manager 51 with an indication whether the task was a "failure" or a "success".

Tasks are interdependent. A virtual system configuration created by task 'A' is used as input to the next task 'A+1' and the virtual system configuration created by task 'A+1' used as input to the next task 'A+2'. Consequently, when a task fails all associated tasks fail as well.

Therefore, task 160 creates virtual configuration 165 that is used as an input to task 161. Task 161 creates virtual configuration 166 that is used as an input to task 162 and so on. After the completion of the last task N+1, an output list of possible upgrade sequences and preferment graphs 170 is displayed to the user of client device 40. Optionally, a real system upgrade 172 is initiated, downloader module 52 downloads the necessary component objects 172 from component objects file 30 of server device 20, installer module 50 installs the requisite component (176) and a new upgraded system 178 is created.

Virtual System Configurator Client system preferably incorporates all the necessary tables and modules a required client system upgrade with dependency checks and automatic conflict resolving in a computing environment based on the Linux operating system.

Persons skilled in the art will appreciate that the present invention is not limited to what has been particularly shown and described hereinabove. Rather the scope of the present invention is defined only by the claims that follow.

CLAIM:

1. A method of supporting computer system management characterized by providing assistance to users of client systems in the conduct of upgrading software components of a computing environment, comprising:

providing a virtual system configurator application program (48) adapted to execute an automatic system upgrade;

providing a software component knowledge base (90) to be used as an information source to enable said system upgrade ;

providing original system information creation application program to be used to formulate an original system information base (46);

providing a storage device (41) that is able to hold said applications and information bases;

providing an I/O device (45) to be used as a Man Machine Interface (MMI) via which said applications are activated;

providing a Central Processing Unit (CPU) (43) operative in the execution of said programs.

2. The method, as claimed in claim 1, characterized by providing said original system configuration application (102) with the capability of building a kernel configuration table (72) to hold information relevant to the existing kernel components in the computing environment.

3. The method, as claimed in claim 1, characterized by providing said original system configuration application (102) with the ability to build a hardware configuration table (74) to hold information relevant to the existing hardware-controlling components in the computing environment.

4. The method, as claimed in claim 1, characterized by providing said original system creation application (102) with the capability of building a software configuration table (76) to hold information relevant to the existing software components in the computing environment.

5. The method, as claimed in claim 1, further characterized by providing said virtual system configurator application (48) to build (116) a system upgrade table (44) to used as a control information base in said system upgrade process. .

6. The method, as claimed in claim 5, further characterized by building said system upgrade table (44) from a predetermined list of new components (92).

7. The method, as claimed in claim 5, further characterized by building said system upgrade table (44) from a predetermined list of improved components (92).

8. The method, as claimed in claim 1, characterized by providing a user preferences setting application (108) to make available said user of said computing environment the option of selecting the number and type of components to upgrade (92).

9. The method, as claimed in claim 8, further characterized by enabling said user via said user preferences setting application (108) to request the automatic upgrading (129) of kernel components.

10. The method, as claimed in claim 8 to 9, further characterized by enabling said user via said user preferences setting application (108) to request automatic notification (130) of new or improved components from the central virtual system configurator server (20).

11. The method, as claimed in claim 8 to 10, further characterized by providing said user the option to request automatic downloading (132) of said new or improved components (30) from said central virtual system configurator server (20).

12. The method, as claimed in claim 1, characterized by making available to said user the option of interrogating (122) said original system information tables (46).

13. The method, as claimed in claim 1, further characterized by providing said user the capability of creating a user profile information table (126)

to enable said central virtual system configuration server (20) to transmit new and improved components (30) as well as notifications, and automatic kernel upgrades according to said user profile table (126).

14. The method, as claimed in claim 1, further characterized by providing said user with the option of sending queries (110) to said central virtual system configuration server (20).

15. The method, as claimed in claim 14, characterized by providing said user with the option of querying component information or component news pertaining to new or improved components (28).

16. The method, as claimed in claim 1, further characterized by providing said user with the option of interrogation said original information tables (122).

17. The method, as claimed in claim 1, further characterized by providing said user with the option of checking for inter-component dependency conflicts (106) regarding the original system information table (46) .

18. The method, as claimed in claim 1, further characterized by providing said user the capability of downloading (112) new and improved component objects and component sources by request.

19. The method, as claimed in claim 1, characterized by making available to said user the option of initiating a virtual system configuration process (48).

20. The method, as claimed in claim 1, further characterized by executing said virtual system configuration process (48) by using said component knowledge base (90).

21. The method, as claimed in claim 1, whereby said component knowledge base (90) comprises:

a version tree table (92) utilized as a source of component information or component data;

an xor rules table (94), which holds the 'xor' rules necessary for a dependency rules checking procedure;

an add-remove rules table (96) which holds 'need' rules necessary for said dependency rules checking procedure;

a links table (98) necessary for the implementation of improved kernel components.

22. The method, as claimed in claim 1, characterized by providing the option of executing said system upgrade in a fully automatic mode.

23. The method as claimed in claim 22, further characterized by providing said user the option of executing said upgrade in a semi- automatic mode.

24. The method, as claimed in claim 22 to 23, further characterized by providing said user the option of a partial execution of said system upgrade.

25. The method, as claimed in 22 to 24, further characterized by making available said user the option of executing a number of system upgrades concurrently (160,161,162,163,164).

26. The method, as claimed in 22 to 25, further characterized by enabling said user to control the concurrently running system upgrades with a task manager application (51).

27. The method, as claimed in claim 1, whereby said virtual system configurator application (48) performs appropriate inter-component dependency checks (54).

28. The method, as claimed in claim 27, whereby said virtual system configurator application (48) resolves automatically all inter-component dependency conflicts.

29. The method, as claimed in claim 27 to 28, whereby said virtual system configurator application (48) resolves automatically all inter-component dependency conflicts unless said user by exercising said user preferences setting option (124) sets contrary preferences regarding said

components involved in said dependency conflicts or said components involved in the solution of said dependency conflicts.

30. The method, as claimed in claim1, whereby said virtual system configuration process (48) outputs a list of alternative actions (170) pertaining to a genuine system upgrade (176) in said computing environment.

31. The method, as claimed in claim 30, whereby said virtual system configuration process (48) output (170) is formulated in a heuristically arranged sequence of alternative actions pertaining to said genuine system upgrade (176) in said computing environment.

32. The method, as claimed in claim 1, whereby said user enabled to perform a genuine system upgrade (176).

33. The method, as claimed in claim 1, where said genuine system upgrade (176) results in an upgraded system (170) in said computing environment.

34. The method, as claimed in claim 1, whereby said user controls the execution of said virtual system configurator application (48) and the related application thereof via commands entered into, and information read from said I/O device (45) via a predefined Man Machine Interface (MMI).

35. A system providing support to computer system management in client system platforms by running a virtual system upgrade process regarding the installation, uninstallation or update of software elements, comprising;

storage means to hold the software components, application programs and application supporting information bases;

an I/O device to be used as interfacing means for a user to input commands therein and reading information therefrom by which to activate said applications and maintain said information bases;

a Central Processing Unit to be used as processing means to enable said applications to execute therein;

virtual system upgrade means such as a virtual system configurator application to be used as a inter-component dependency checker and conflict resolver;

original system configuration application means to be used in creating original system configuration tables;

original system configuration and component knowledge means such as tables and information bases to be utilized as data controlling the execution of said virtual system upgrade means.

36. The system, as claimed in claim 35, characterized in that said original system configuration means have the capability of building a kernel components configuration table, a hardware-controlling components configuration table, and a software components configuration table.

37. The system, as claimed in claim 35, characterized in that said virtual system upgrade application is enabled to build a system upgrade table for the purposes of controlling the dependency checking and conflict resolving activities of the virtual system upgrade.

38. The system, as claimed in claim 37, characterized in that said system upgrade table is built by said user from a predetermined list of new components.

39. The system, as claimed in claim 37, characterized in that said system upgrade table is created by said user from said predetermined list of improved components.

40. The system, as claimed in claim 35, characterized in that a user preferences setting means is provided to make available to said the option of determining the components to be upgraded in the preferred virtual system upgrade process.

41. The system, as claimed in claim 40, further characterized in that said user preferences setting means makes available said user the option of requesting automatic upgrading of said kernel components.

42. The system, as claimed in claim 40, further characterized in that said user preferences setting means makes available said user the option of requesting automatic notification about new and improved components from the central virtual system configurator server.

43. The system, as claimed in claim 40 to 42, further characterized in that the user is enabled to request automatic transmission of new and improved components from said central server to said client system.

44. The system, as claimed in claim 35, further characterized by providing querying means (122) by which said user can interrogate said original system configuration tables (46).

45. The system, as claimed in claim 35, characterized in that user profile creating means are provided to enable said central virtual system configurator server (20) to download to said client system platform all the requested information and objects related to said client profile.

46. The system, as claimed in claim 35, characterized in that the central virtual system configuration server is providing component information according to the demands of said user of said client system.

47. The system, as claimed in claim 35, characterized in that said user is provided with original system configuration browsing means to be used as dependency checks regarding components installed in the original system.

48. The system, as claimed in claim 35, characterized by downloading means provided to said user for the purpose of receiving from said central server new components or improved components including component sources and component news.

49. The system, as claimed in claim 35 characterized by providing initiating means by which said user is enabled to exercise the option of executing a virtual system upgrade process.

50. The system, as claimed in claim 35, characterized in that said user is provided with various options in executing said virtual system upgrade process.

51. The system, as claimed in claim 50, whereby said user enabled to execute said virtual system upgrade process in automatic or semi-automatic mode.

52. The system, as claimed in claim 50 to 51, whereby said user provided the capability to execute only sections of said virtual system upgrade process.

53. The system, as claimed in claim 50 to 52, further comprising means to enable said user to run a plurality of virtual system upgrades simultaneously.

54. The system, as claimed in claim 50 to 53, whereby said means to run said plurality of said virtual system upgrades is a task manager system.

55. The system, as claimed in claim 35, characterized by said virtual system upgrade system is having dependency checking means and inter-component dependency conflict resolving means.

56. The system, as claimed in claim 55, whereby said inter-component dependency conflict resolving means is automatic.

57. The system, as claimed in claim 55 to 56, characterized in that said dependency checking means and dependency conflict resolving means are incorporated into a specific conflict resolver module.

58. The system, as claimed in claim 35, characterized in providing output means whereby a list of alternative actions is created to assist said user in the execution of a genuine system upgrade.

59. The system, as claimed in claim 58, characterized in that said list of alternative actions is sorted in order of heuristically calculated preferred choices.

60. The system, as claimed in claim 35, characterized in that said user is provided genuine system upgrade means through which one of the actions on the list of alternative actions would be executed in order to accomplish a preferred genuine system upgrade.

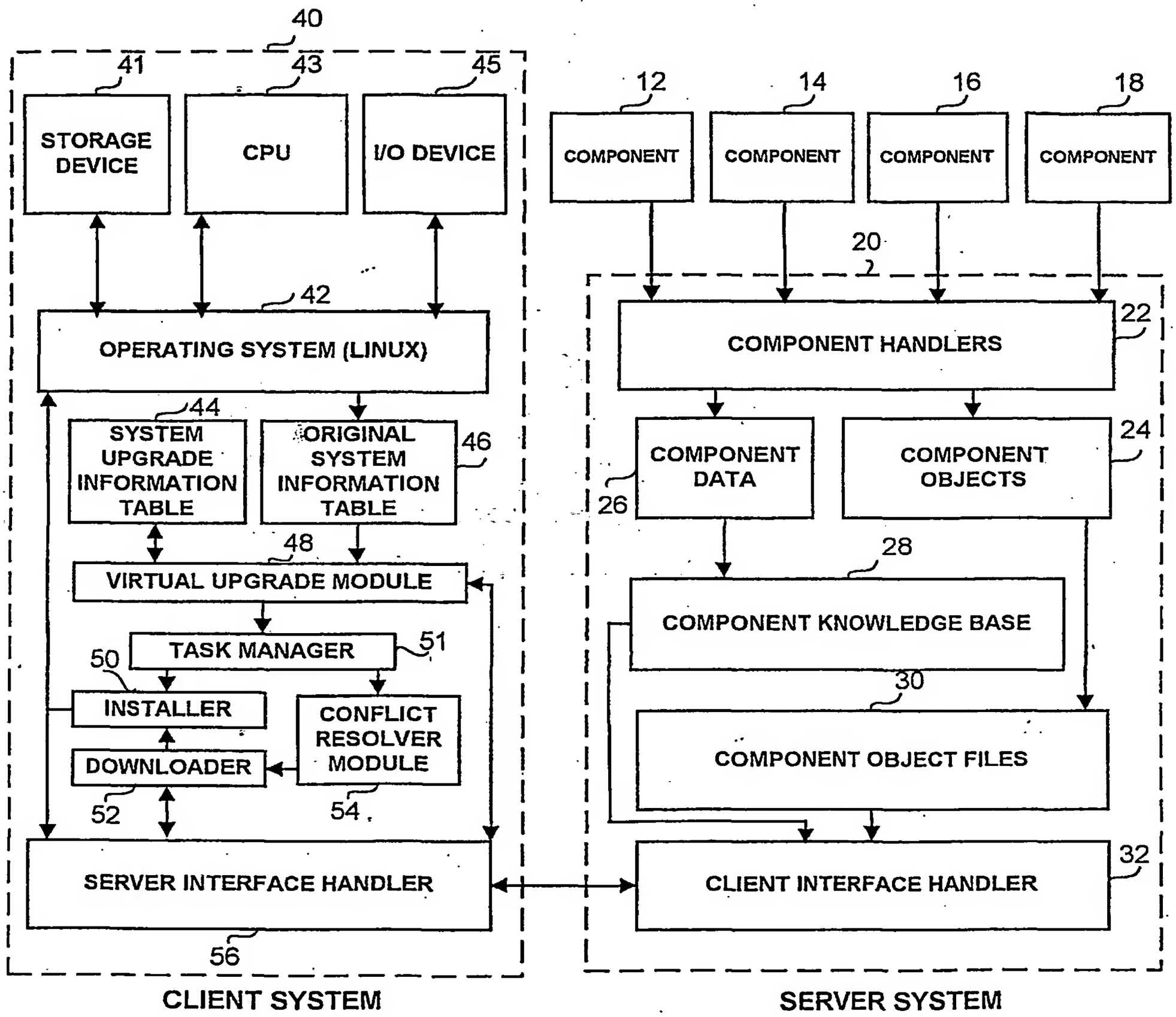

40
41
43
45
STORAGE DEVICE
CPU
I/O DEVICE
12
14
16
18
COMPONENT
COMPONENT
COMPONENT
COMPONENT
20
42
OPERATING SYSTEM (LINUX)
22
COMPONENT HANDLERS
44
SYSTEM UPGRADE INFORMATION TABLE
ORIGINAL SYSTEM INFORMATION TABLE
46
COMPONENT DATA
26
COMPONENT OBJECTS
24
48
VIRTUAL UPGRADE MODULE
28
COMPONENT KNOWLEDGE BASE
TASK MANAGER
51
50
30
INSTALLER
CONFLICT RESOLVER MODULE
COMPONENT OBJECT FILES
DOWNLOADER
52
54
SERVER INTERFACE HANDLER
32
CLIENT INTERFACE HANDLER
56
CLIENT SYSTEM
SERVER SYSTEM

FIG. 1

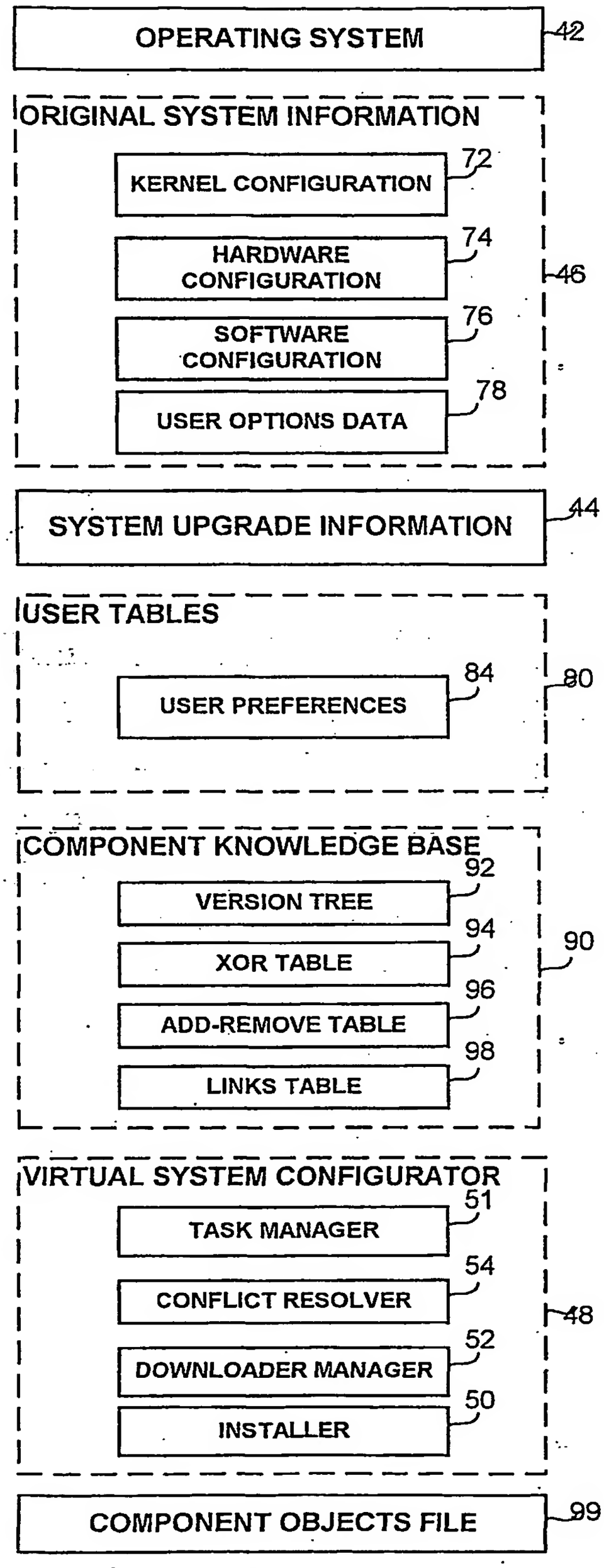

OPERATING SYSTEM
42
ORIGINAL SYSTEM INFORMATION
KERNEL CONFIGURATION
72
HARDWARE CONFIGURATION
74
SOFTWARE CONFIGURATION
76
USER OPTIONS DATA
78
46
SYSTEM UPGRADE INFORMATION
44
USER TABLES
USER PREFERENCES
84
80
COMPONENT KNOWLEDGE BASE
VERSION TREE
92
XOR TABLE
94
ADD-REMOVE TABLE
96
LINKS TABLE
98
90
VIRTUAL SYSTEM CONFIGURATOR
TASK MANAGER
51
CONFLICT RESOLVER
54
DOWNLOADER MANAGER
52
INSTALLER
50
48
COMPONENT OBJECTS FILE
99

FIG. 2

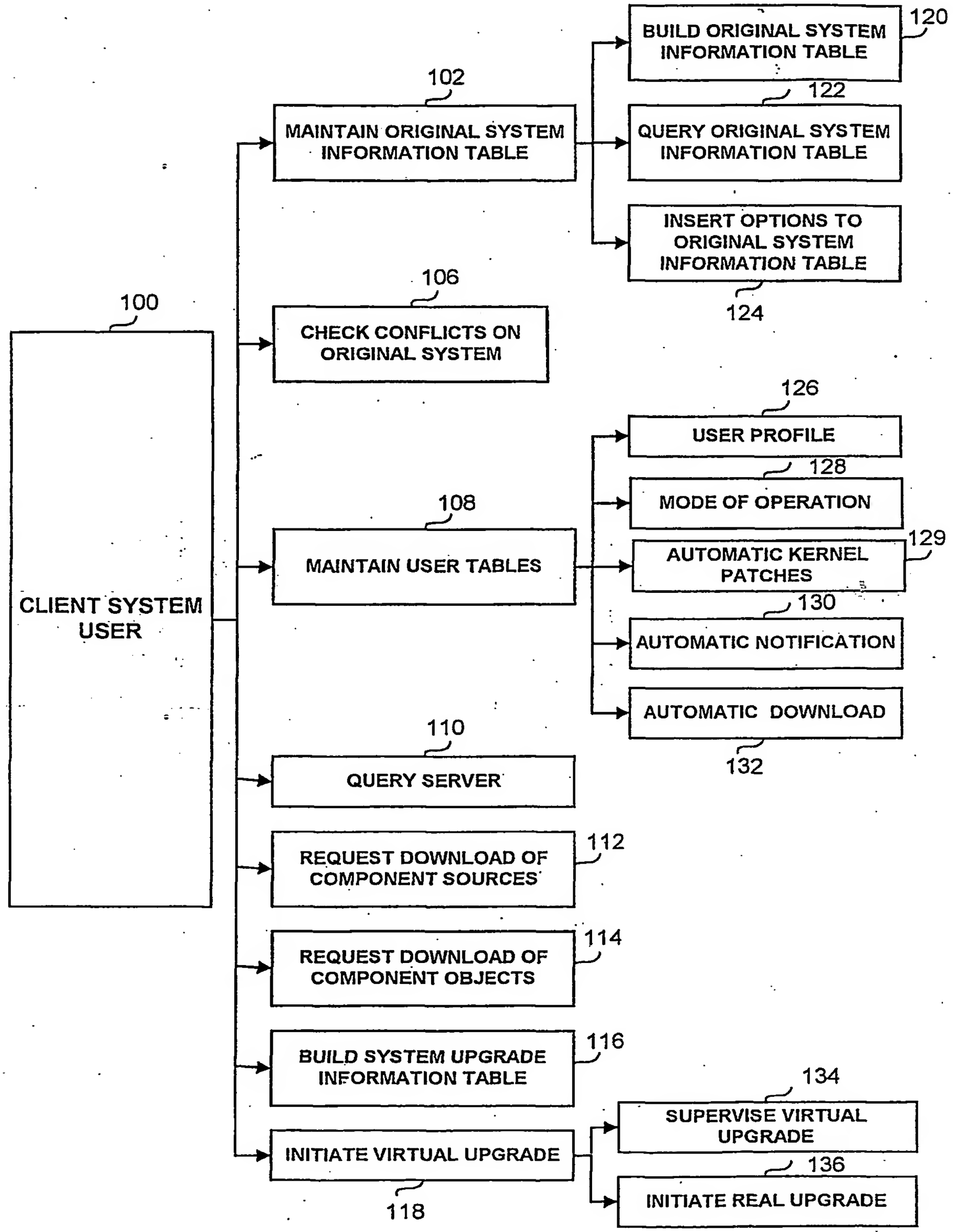

100
CLIENT SYSTEM USER
102
MAINTAIN ORIGINAL SYSTEM INFORMATION TABLE
120
BUILD ORIGINAL SYSTEM INFORMATION TABLE
122
QUERY ORIGINAL SYSTEM INFORMATION TABLE
INSERT OPTIONS TO ORIGINAL SYSTEM INFORMATION TABLE
124
106
CHECK CONFLICTS ON ORIGINAL SYSTEM
126
USER PROFILE
128
MODE OF OPERATION
108
MAINTAIN USER TABLES
129
AUTOMATIC KERNEL PATCHES
130
AUTOMATIC NOTIFICATION
AUTOMATIC DOWNLOAD
132
110
QUERY SERVER
112
REQUEST DOWNLOAD OF COMPONENT SOURCES
114
REQUEST DOWNLOAD OF COMPONENT OBJECTS
116
BUILD SYSTEM UPGRADE INFORMATION TABLE
134
SUPERVISE VIRTUAL UPGRADE
136
INITIATE REAL UPGRADE
INITIATE VIRTUAL UPGRADE
118

FIG. 3

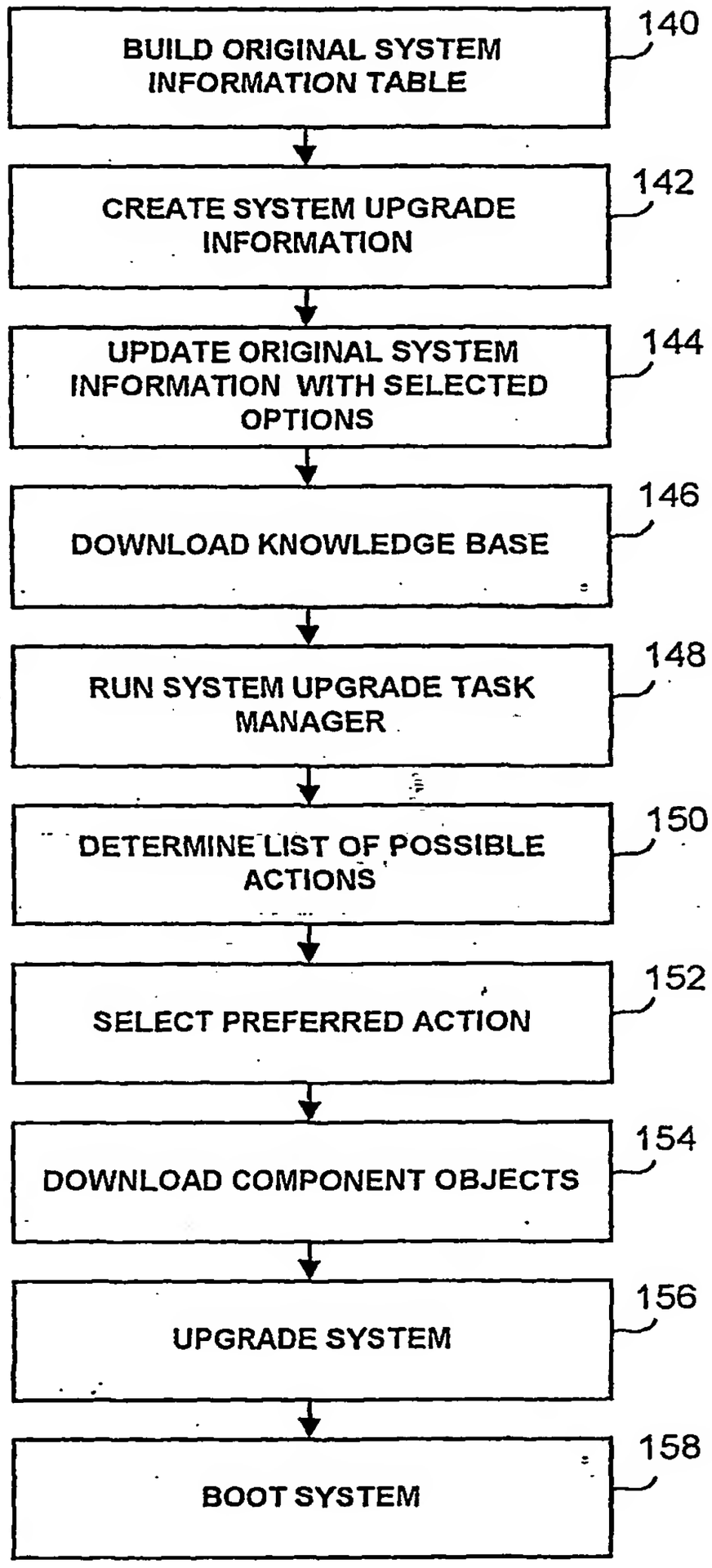

BUILD ORIGINAL SYSTEM INFORMATION TABLE
140
CREATE SYSTEM UPGRADE INFORMATION
142
UPDATE ORIGINAL SYSTEM INFORMATION WITH SELECTED OPTIONS
144
DOWNLOAD KNOWLEDGE BASE
146
RUN SYSTEM UPGRADE TASK MANAGER
148
DETERMINE LIST OF POSSIBLE ACTIONS
150
SELECT PREFERRED ACTION
152
DOWNLOAD COMPONENT OBJECTS
154
UPGRADE SYSTEM
156
BOOT SYSTEM
158

FIG. 4

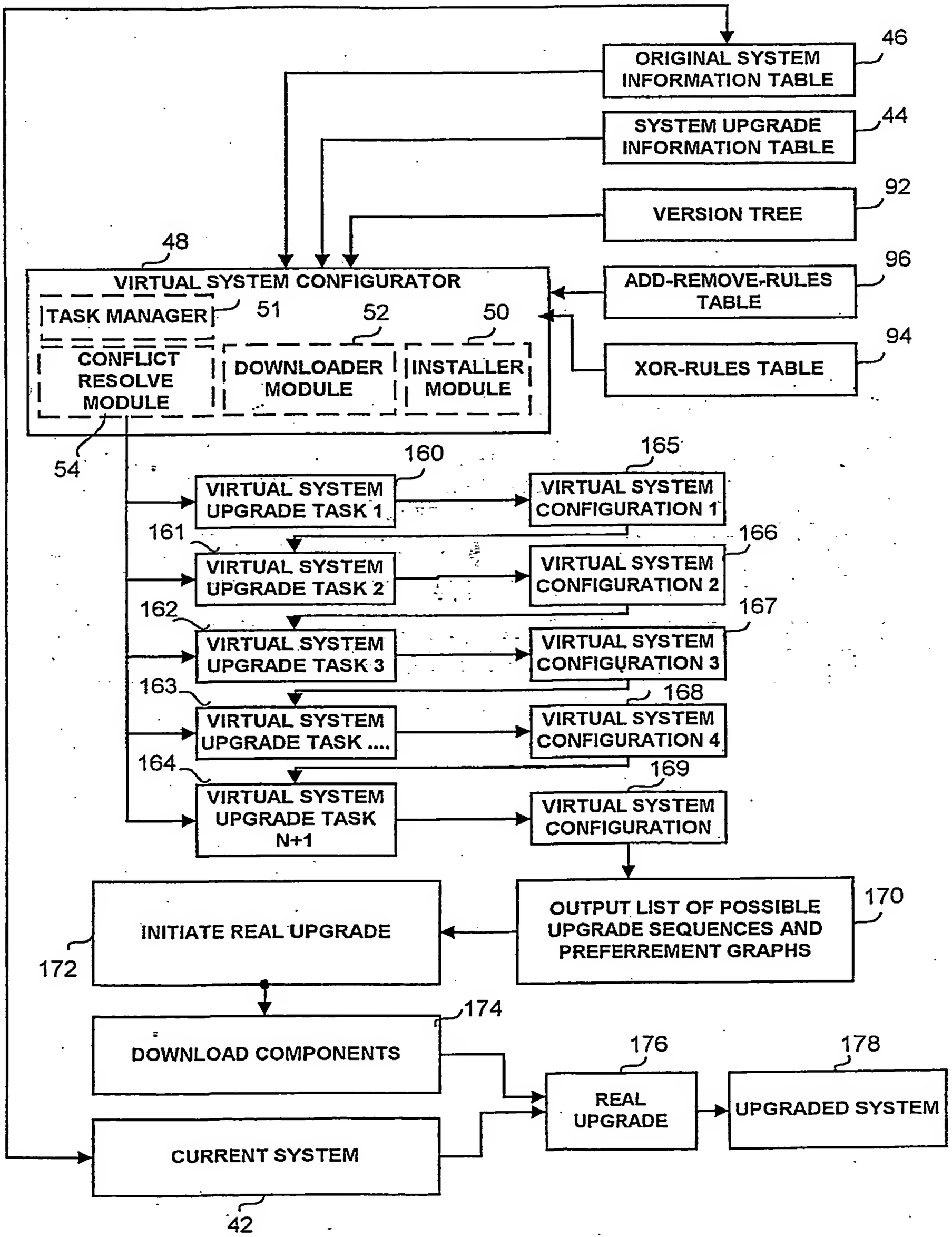

ORIGINAL SYSTEM INFORMATION TABLE
46
SYSTEM UPGRADE INFORMATION TABLE
44
VERSION TREE
92
ADD-REMOVE-RULES TABLE
96
XOR-RULES TABLE
94
48
VIRTUAL SYSTEM CONFIGURATOR
TASK MANAGER
51
52
50
CONFLICT RESOLVE MODULE
DOWNLOADER MODULE
INSTALLER MODULE
54
160
VIRTUAL SYSTEM UPGRADE TASK 1
165
VIRTUAL SYSTEM CONFIGURATION 1
161
VIRTUAL SYSTEM UPGRADE TASK 2
166
VIRTUAL SYSTEM CONFIGURATION 2
162
VIRTUAL SYSTEM UPGRADE TASK 3
167
VIRTUAL SYSTEM CONFIGURATION 3
163
VIRTUAL SYSTEM UPGRADE TASK ....
168
VIRTUAL SYSTEM CONFIGURATION 4
164
VIRTUAL SYSTEM UPGRADE TASK N+1
169
VIRTUAL SYSTEM CONFIGURATION
OUTPUT LIST OF POSSIBLE UPGRADE SEQUENCES AND PREFERREMENT GRAPHS
170
INITIATE REAL UPGRADE
172
DOWNLOAD COMPONENTS
174
REAL UPGRADE
176
UPGRADED SYSTEM
178
CURRENT SYSTEM
42

FIG. 5